Integrating Web Services with Agent Based Workflow Management System (WfMS)

Bastin Tony Roy Savarimuthu, Maryam Purvis, Martin Purvis and Stephen Cranefield
Department of Information Science, University of Otago, Dunedin, New Zealand
{tonyr,tehrany,mpurvis,scranefield}@infoscience.otago.ac.nz}

Abstract

Rapid changes in the business environment call for more flexible and adaptive workflow systems. Researchers have proposed that Workflow Management Systems (WfMSs) comprising multiple agents can provide these capabilities. We have developed a multi-agent based workflow system, JBees, which supports distributed process models and the adaptability of executing processes. Modern workflow systems should also have the flexibility to integrate available web services as they are updated. In this paper we discuss how our agent-based architecture can be used to bind and access web services in the context of executing a workflow process model. We use an example from the diamond processing industry to show how our agent architecture can be used to integrate web services with WfMSs.

1. Introduction

Workflow management systems (WfMSs) [7, 9] are widely used to manage business processes due to their known benefits such as automation, coordination and collaboration between entities. Still, the existing, commercially available workflow management systems do not offer sufficient flexibility for distributed organizations participating in the global market. Existing systems have rigid, centralized architectures that do not operate across multiple platforms. Improvements can be made to a WfMS by employing a distributed network of autonomous software agents that can adapt to changing circumstances. In particular some of the reasons for wanting an adaptive WfMS are:

• It may not be possible to specify all the process details associated with a complex process at the outset. The initial model may represent a high-level view of the process, which includes some of the sub processes. Gradually some of these sub processes may be refined as the stakeholders obtain more experience and knowledge of a particular process.

• Due to changes in the market or regulatory environment, new requirements may be imposed which can impact the process definition. Changes in the market may also include the availability of some new technologies and new services such as web services, which may require the modification of the process.

The work of Fleurke et al. [4] deals with the framework of a distributed network of autonomous software agents that can adapt to the changing circumstances in a workflow management system. The business processes undergo changes over time to accommodate a changing environment such as the availability of web services. Business processes should be able to take advantage of web services that are available in an intranet as well as in the Internet, such as stock monitoring web services. The workflow system that models these business processes should have necessary mechanisms to integrate and use these web services. In this paper we describe the extension to the work done by Fleurke et al [4]. The enhanced framework provides mechanisms to create agents that are capable of accessing various web services.

The paper is organized as follows. The next section gives an overview of the background. The architecture of the system is described in Section 3. In the fourth section we discuss the underlying mechanism of integrating web services with our workflow system based on multiple agents. We present the conclusions and future directions of our work in section five.

2. Background

In this section we explain the background of our work, which includes the Coloured Petri nets that are used to design the process models, the multi-agent system on which our workflow system has been built, and some prior approaches in using web services with agents.

2.1. Coloured Petri Nets (CPNs)

We use CPN as a formalism to model workflows in our system. The sound mathematical foundation behind the Coloured Petri nets (CPNs) makes it a very useful tool for modeling distributed systems. Petri nets consist of four basic elements. The *tokens* which are typed markers with values, the *places* that are typed locations that can contain zero or more tokens, the *transitions* which

represent actions whose occurrence can change the number, locations and value of tokens in one or more of the places connected to them and the *arcs* that connect places and transitions. A detailed description of CPNs can be found in [6].

2.2 WfMS driven by agents

Multi agent systems offer distributed and open platform architecture and hence can support dynamically changing systems. Our WfMS is partitioned among various interacting agents following the interaction protocols. The model associated with a business process is represented with Coloured Petri net formalism that is executed by a specially designed agent. This agent-based environment facilitates the dynamic incorporation of changed models in the system and thereby assists process re-engineering. Advantages of employing agents include the facilitation of inter and intra organizational co-operation and flexibility in process determination and resource utilization.

2.2. Web Services

Web Services are software components available on the Internet, which provide certain services that may be of general interest, such as weather monitoring services, currency converters, etc. A large fraction of the web services are used within companies protected within their own firewalls. These web services can be accessed for day-to-day business transactions. Examples of these web services include banking services and air ticket booking. The workflow process modeler can integrate web services with the existing workflow system. For example, a process model associated with the travel plan of a tourist may depend upon environmental factors, such as the weather conditions. The task associated with finding the weather condition can be provided using a web service.

2.3. Related Work

Some researchers have integrated agent-based workflow systems with web services [3]. However enhancements can be made to improve these approaches. In the research done by Buhler et al. [3], BPEL4WS [1] has been used as a process model and this model is converted to a Petri net. The problem with this approach as acknowledged by the authors, is that the demonstration system developed by the researchers so far does not support some of the simple constructs of BPEL4WS. However, in our system the process model is described using a coloured Petri net that can be directly executed. Our system does not require the conversion of a BPEL process into a Petri net process. The conversion mechanism of a BPEL4WS model to a Petri net model has to be validated to ensure the structural and behavioural equivalence associated with the original model.

3. System Architecture

Our system is based on the FIPA [2] compliant agent platform, Opal and uses the CPN execution tool JFern [8]. Our system consists of seven types of special Opal agents, which provide the functionality to control the workflow. The manager agent provides all functionality the workflow manager needs, such as creation and deletion of tasks, roles and process definitions, instantiation of new process instances, and creation of resource agents. The process agent executes a process instance. Each resource in the system has its own resource agent. Every resource in the system gets registered to one of the broker agents that allocate resources to the process. The storage agent manages the persistent data that is needed. The monitor agent collects all the process-specific data and sends them to the storage agent. The control agent continuously looks for anomalies to the criteria specified by the human manager and reports the violations of these criteria to the manager agent. The manager agent provides information to the human manager, which can be used as a feedback mechanism. A detailed description of these agents can be found in our previous work [5].

4. Integrating web services

In our design we have wrapped the web service as an agent. A specialized agent called Web Service Agent (WSA) is created, and it can be used to query various operations exposed by the web service.

4.1 Demonstration of agent-based integration of web services using an example from diamond processing industry

In this section we describe how a web service has been integrated with the multi-agent based workflow management system. The Petri net model shown in figure 1 illustrates how agents can be used in a diamond processing industry. When the process agent executes the Petri net model, a token representing a job is created at the *Start* node. The attributes encoded in the job token in the form of name-value pairs are:

- Attribute 1: (URIName, http://www.stardiamonds.com/diamonds.wsdl)
- Attribute 2: (connectAndQueryWebService, getAllStoneDetails)
- Attribute 3: (determinePrice, getPrice)

The process agent assigns to the resource agent the task of creating a web service agent as inscribed in *createWebServiceAgent* task definition. The resource agent accesses the URI from the job token and reads the WSDL [10] document and uses wsdl2java from the Axis toolkit [11] to create the necessary stubs.

The resource agent sends a message to the process agent about the creation of the Web Service Agent, which is capable of handling requests for operations defined in WSDL. Once the process agent receives a message from the resource agent that a web service agent has been successfully created, the process agent executes the next transition *connectAndQueryWebService*. The operation that is to be invoked is encoded in the job token that is passed along when a transition has been fired successfully. The process agent assigns this task to the web service agent by sending the message to the web service agent with the details about the operation that is to be invoked. The web service agent invokes the web service, obtains the result, and sends back the result to the process agent as a message. The results are stored in the job token.

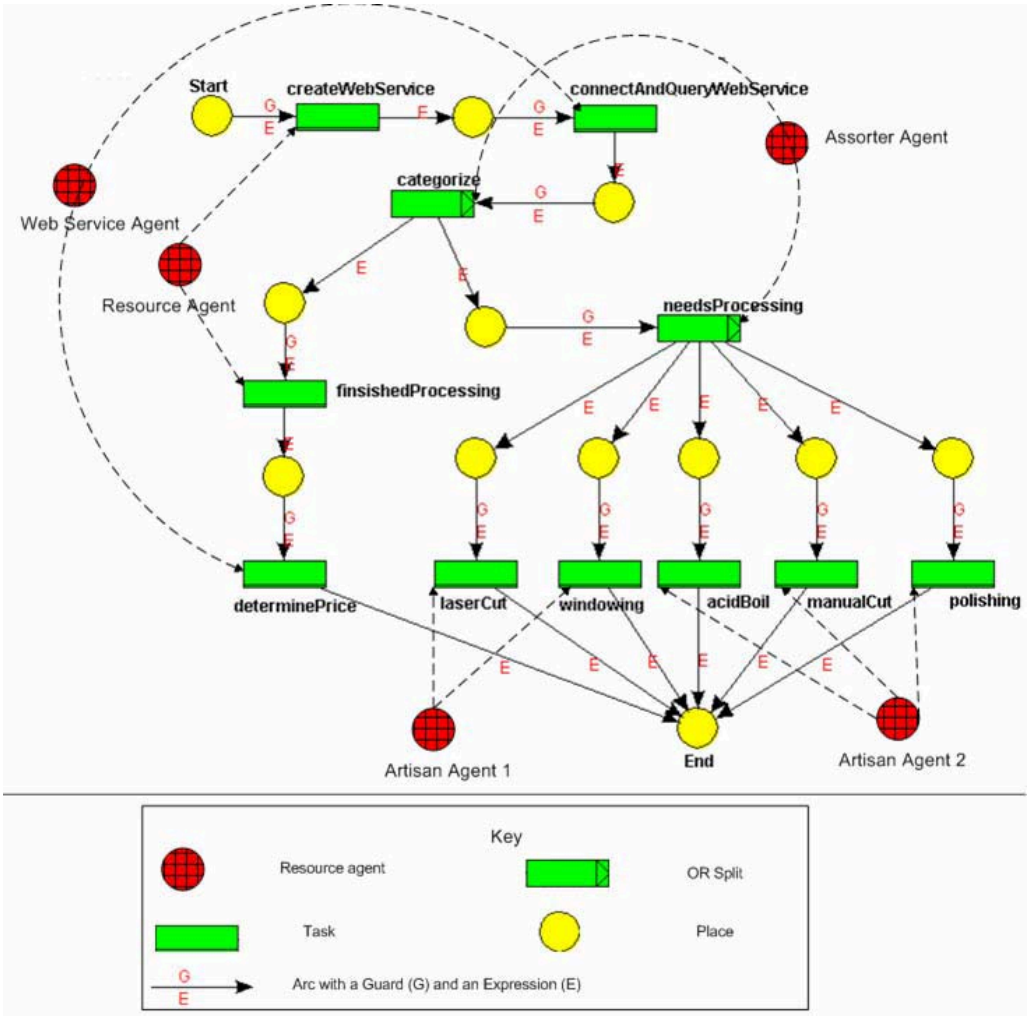

Fig.1 A snapshot of the diamond processing process model during execution

The result consists of an array comprising of details of the diamonds. For each diamond in the array, a job token is created. A diamond job token would contain details such as *clarity, color, cut, carat weight, lusture, nextProcess, nextArtisan, hasProcessingFinished, price* and name-value pairs of the web service operations that are to be invoked. The name represents the transition that is executed and the value represents the operation that is

called on the web service. These job tokens are available for the assorter agent[1] who decides if the diamond has been processed. The process agent directs the assorter agent to decide whether each of the diamonds requires further processing. If a diamond requires further processing, the assorter agent determines which process the diamond should undergo and then allocates the appropriate artisan agent to perform that task. If the assorter agent decides that the diamond has been processed completely, it updates the details of the diamond token that can be sent to the evaluator. This is done be setting a value for the boolean attribute *hasProcessingFinished* for that particular token to indicate whether the diamond has been processed. The web service agent, which acts as a evaluator determines the price of the token depending upon the 4 c's of diamond quality, namely *color, clarity, cut, carat weight*, and assigns a price for that diamond. Each artisan agent is capable of performing one or more tasks. These tasks are *lasterCut, windowing, acidBoil, manualCut* and *polishing*. Each unprocessed diamond is assigned to one of these tasks by the assorter agent. Depending upon the attribute of the job token, these diamonds will be available to a particular artisan agent. The artisan agent performs a particular task on the diamond. After the completion of a task, the artisan agent requests the process agent to add a name-value pair to the attribute list of the job token to indicate the completion of the task. For example, after the completion of a task named *laserCut*, an attribute binding such as (*laserCut*, true) is added to the job token.

After a stone has been processed into a diamond that can be sold, it is evaluated and the diamond is made available at the end place for further operations such as *dispatchToHeadOffice*. If the processing of the diamond is not complete, the stone details can be obtained from the diamond processing web service. The diamond goes through an iteration of several processes before it is ready for the market. After every task is executed, the storage agent persistently stores all the details of a diamond. Figure 5 also shows the pool of resource agents and the process model that is executed. The dotted arrows start from an agent and end in a transition of the CPN. This indicates that the agent is able to perform the task indicated by the transition.

5. Conclusions and future work

We have described our flexible, agent based architecture for workflow management systems. The agent-based architecture facilitates the easy integration of Web Services with the workflow system. We have demonstrated using an example how a web service can be used in a diamond processing workflow with limited effort. The web service agent will be able to connect to any web service dynamically. We are currently extending our architecture to accommodate a process model that executes composite web services. A full version of our paper can be found in [12].

[1] The job description of an assorter is defined at http://www.occupationalinfo.org/77/770281010.html

6. References

[1] Business Process Execution Language for Web Services (BPEL4WS). http://www.bpelsource.com

[2] Foundation for Intelligent Physical Agents (FIPA). http://www.fipa.org.

[3] Paul Buhler and Jose M. Vidal. Enacting BPEL4WS specified workflows with multi-agent systems. In Proceedings of the Workshop on Web Services and Agent-Based Engineering, 2004.

[4] Martin Fleurke, Lars Ehrler, and Maryam Purvis. JBees – an adaptive and distributed framework for workflow systems. In Workshop on Collaboration Agents: Autonomous Agents for Collaborative Environments (COLA), Halifax, Canada, pages 69–76, 2003.

[5] Savarimuthu, BTR and Purvis, M. and Fleurke, M. Monitoring and controlling of a workflow management system. In Proc. Australasian Workshop on Data Mining and Web Intelligence (DMWI2004), pages 127–132.

[6] K. Jensen. Coloured Petri Nets - Basic Concepts, Analysis Methods and Practical Use, Volume 1: Basic Concepts. EATCS Monographs on Theoretical Computer Science. Springer-Verlag, 1992.

[7] S. Meilin, Y. Guangxin, X. Yong, and W Shangguang. Workflow Management Systems: A Survey. In Proceedings of IEEE International Conference on Communication Technology (ICCT 1998), ISBN 0-7803-5156-9, 1998.

[8] JFern - http://sourceforge.net/projects/jfern/, 2003.

[9] W.M.P van der Aalst and K. van Hee. Workflow Management: Models, Methods, and Systems. MIT Press, 2002.

[10] Web Service Definition Language, www.w3.org/TR/wsdl

[11] Apache Axis Toolkit. http://ws.apache.org/axis/

[12] Savarimuthu, BTR., Purvis, M.A., Purvis, M.K., and Cranefield, S., "Agent-Based Integration of Web Services with Workflow Management Systems", Information Science Discussion Paper Series, Number 2005/05, ISSN 1172-6024, University of Otago, Dunedin, New Zealand (2005).